MACHINE LEARNING FOR FRAUD DETECTION IN FINANCIAL TRANSACTIONS

A PROJECT REPORT

Submitted by

A. D. Thushani Niwarthana - 137186

A. D. Wishani Samadara - 137187

Lubabatu Hussaini - 137188

in partial fulfillment for the award of the degree of

Bachelor of Science in Computer Science

IN

Department of Computer Science

NIMS University

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report **"Machine Learning For Fraud Detection In Financial Transactions"** is the bonafide work of "A.D. Thushani Niwarthana, A.D. Wishani Samadara, Lubabatu Hussaini" who carried out the project work under supervision.

SIGNATURE **SIGNATURE**

HEAD OF THE DEPARTMENT **SUPERVISOR**

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

1. Table of Contents

2. List of Table

3. List of Figure

4. Abstract

The increasing fraudulent activities with the development of online transactions cause a significant challenge to financial institutions. Therefore, the present Fraud Detection System has utilized advanced machine-learning algorithms to classify fraud alerts from others actual legitimate transactions. A well-curated dataset is subjected to extensive preprocessing ensuring that quality and relevance are derived to devise effective models.

Various machine learning algorithms such as Random Forest, Support Vector Machines, and Gradient Boosting are being used for training the predictive model along with its performance on a separate testing dataset. The results show that the system can classify the legitimacy of transactions well with an accuracy rate of 76.67%. Therefore, it acts as a clear indicator of what the model can do in terms of identifying legitimacy or otherwise of a transaction.

The system offers a user-friendly interface to input transaction details for receiving immediate predictions, thereby making the model useful in practical scenarios. The project, thereby, demonstrates the ability of machine learning to prevent fraud and elaborates the point that ongoing improvement and changes are essential against the evolving fraudulent methodologies.

Future work would involve increasing the ability of the system to enhance its capability through integrating sophisticated algorithms and deep learning techniques. Other ensemble methods will be implemented to have an improvement in quality of detection and a reduction in false positives. Another avenue could involve real-time data feeds and user feedback mechanisms to construct an even more dynamic and responsive fraud detection system.

5. Introduction

5.1 Identification of the Problem

As digital transactions increase, so do banking and financial frauds. Such frauds are not only capital losses but also damage the reputation of such financial institutions. Hence, the present need for the real-time detection of fraudulent transactions in online banking services is essential for loss minimization and for preventing customer mistrust.

5.2 Objectives

- To design a suitable machine learning algorithm that could classify transactions as valid and fraudulent.
- To be executed using the Random Forest Classifier as it is robust and efficient in handling large datasets with good functionality in classification tasks.
- Evaluate effectiveness of feature engineering by discussing how different techniques are there for feature Mechanisms for real-time fraud detection to upgrade with real-time fraud detection functionalities to enable immediate transaction monitoring and alerting.
- Conducting a comparative analysis of Machine Learning Algorithms to identify the strengths and weaknesses of each algorithm regarding fraud detection to guide the selection of the best model suitable for deployment.

5.3 Significance of the Project

This project would be crucial in further development in fraud detection methods, and eventually help financial institutions save resources and prevent financial losses. Machine learning models that are optimized for fraud detection would be built through this project, and thus lead to even more secure financial transactions, consumer confidence, and reduces the effects of fraud in the economy. Moreover, it provides a guideline in dealing with common issues in fraud detection, such as data imbalance and adaptive learning, which are relevant to any domain of machine learning.

6. Literature Survey

6.1 Introduction of Fraud Detection Techniques.

- In general, fraud detection techniques could be broadly classified into three categories:

1. Rule-Based Systems - predefined rules and thresholds capture fraud.

2. Statistical Methods - relied upon to identify anomalies

3. Machine Learning Approaches - utilize algorithms trained on historical data to learn the patterns and make predictions.

6.2 Machine Learning in Fraud Detection

Machine learning is the most popular approach used for fraud detection in present due to its ability to analyze massive volumes and intricate patterns in data. Some of the algorithms that have been very successful in fraud detection applications include random forest, decision trees, neural networks, and ensemble methods.

6.3 Random Forest Classifier

In this project Random Forest Classifier is used since it is an ensemble technique for learning which trains multiple decision trees in the process and outputs the mode of their predictions. It is particularly useful for applications involving imbalanced data, a common feature of fraud detection datasets where fraudulent transactions are highly uncommon when compared to valid ones.

6.4. Advantages and Disadvantages of Random Forest Classifier

Table 1 : Advantages and Disadvantages of Random Forest Classifier

Advantages	Disadvantages
The model provide high accuracy due to the ensemble method that combines multiple decision trees, reducing overfitting.	The model can be complex, making it less interpretable compared to simpler models like logistic regression.
Provides insights into feature importance, helping to identify which features contribute most to the prediction, aiding in feature selection.	The performance of the model can depend significantly on hyperparameters, requiring careful tuning to achieve optimal results.
The averaging of multiple trees helps to mitigate overfitting, making the model robust to noise in the data.	Training can be computationally intensive, especially with a large number of trees and features, requiring more memory and processing power.
Can be used for both classification and regression tasks, making it a versatile choice for various applications.	May not perform as well on limited datasets, where many features are irrelevant or contain a lot of missing values.
Random Forest can handle imbalanced datasets effectively by adjusting class weights or through the nature of the ensemble learning.	The training time can be longer compared to single decision trees, particularly when the dataset is large or when many trees are used.

7. Design Flow / Process

7.1 Data Collection

The dataset used in this project is a CSV file obtained from kaggle.com which has a vast collection of data science community with powerful tools and resources. The CSV file contains transaction details with various features, including transaction amount, time, and amount, along with a 'Class' column indicating whether the transaction is fraudulent (1) or valid (0).

7.2 Data Preprocessing

Data preprocessing involves:

- Checking for missing values and handling them appropriately.
- Normalizing numerical features to ensure they are on a similar scale.
- Encoding categorical variables to convert them into a format suitable for model training.

7.3 Flowchart

A flowchart visually represents the sequence of steps or decisions needed to complete a process. In the context of a fraud detection model, the flowchart can illustrate the steps from data collection to model deployment and monitoring.

The below flowchart describes the process of building and using a machine learning model to predict fraudulent transactions:

1. Begins with importing a dataset that includes details about past transactions, including whether they were fraudulent or valid.
2. The dataset is then cleaned and preprocessed, ensuring that it is in the right format for the machine learning algorithm.
3. The model is then trained on the dataset, using the features of the transactions to predict their fraudulence.
4. After the model is trained, its accuracy is evaluated, meaning checking how well the model can predict fraudulence for new, unseen data.
5. The flowchart continues by explaining how to get user input for new transactions and use the trained model to predict whether they are fraudulent or not.
6. The process also includes steps to ensure that the user input matches the expected format for the model.
7. Finally, the prediction result is displayed to the user.

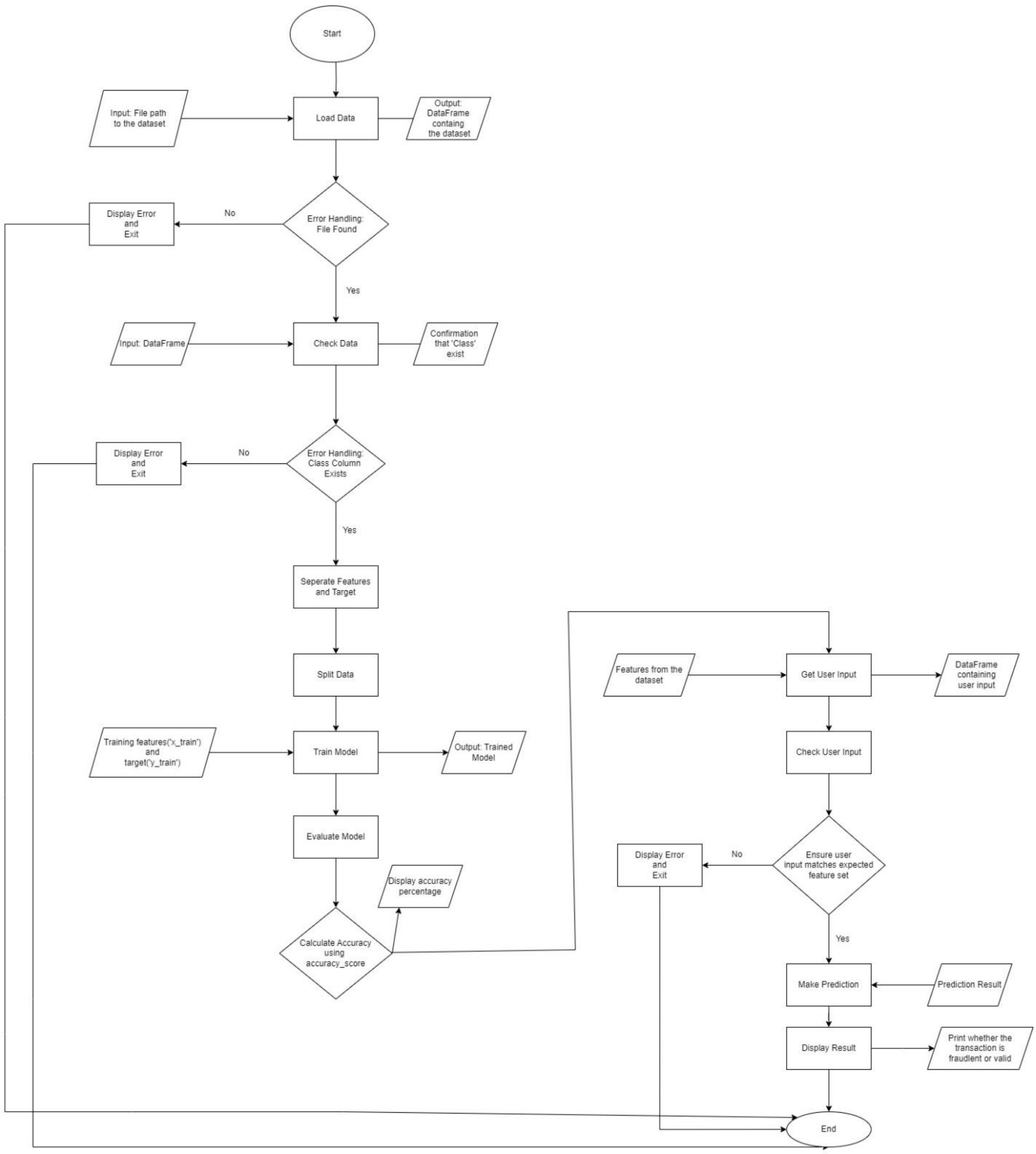

Figure 1: Flowchart

7.4 Exploratory Data Analysis (EDA)

EDA allows for a comprehensive examination of the dataset, helping to identify the underlying structure, patterns, and relationships within the data. It is a crucial step in understanding the dataset and its characteristics. By visualizing distributions, correlations, and trends, one can gain insights into the characteristics of legitimate versus fraudulent transactions.

Various techniques to visualize and summarize the data:

1. Descriptive Statistics:

- Calculating mean, median, mode, standard deviation, and range for numerical features to understand their distributions.

2. Visualizations:

- Histograms were used to visualize the distribution of continuous features, helping to identify skewness and outliers.

- Box plots were utilized to detect outliers and understand the spread of the data.

- Correlation Matrices were created to identify relationships between features, which can be useful for feature selection.

3. Class Distribution:

- Analyzing the distribution of the target variable ('Class') to understand the balance between fraudulent and valid transactions. This is important for selecting appropriate evaluation metrics and model strategies.

7.5 Model Training

Once the data was preprocessed and analyzed, the next step was to train the Random Forest Classifier.

The training process included:

- **Data Splitting:** The dataset was split into training and testing sets using the ***train_test_split*** function from ***sklearn.model_selection***. A typical split ratio of 70% for training and 30% for testing was used. This ensures that the model is trained on a substantial amount of data while retaining a portion for evaluation.

- **Model Initialization:** The Random Forest model was initialized with 100 estimators (trees) and no maximum depth, allowing the trees to grow to their full depth. This helps capture complex patterns in the data.

- **Model Fitting:** The model was trained using the training dataset with the ***'fit'*** method. This involves creating multiple decision trees and aggregating their predictions.

```python

def train_model(X, y):
    # Training the model.
    model = RandomForestClassifier(n_estimators=100, max_depth=None, random_state=42)
    model.fit(X, y)
    return model

```

Figure 2: Model Training

7.6 Model Evaluation

After training the model, it was essential to evaluate its performance on the test dataset:

- **Predictions:** The model was used to predict the classes of the test dataset using the predict method.
- **Accuracy Score:** The accuracy of the model was calculated using the ***'accuracy_score'*** function from ***'sklearn.metrics***'. This metric provides a straightforward measure of how many transactions were correctly classified.
- **Confusion Matrix:** A confusion matrix was generated to visualize the performance of the model. It shows the true positives, true negatives, false positives, and false negatives, providing insights into the model's strengths and weaknesses.
- **Evaluation Metrics:** In addition to accuracy, other metrics such as precision, recall, and F1-score were calculated to provide a more comprehensive evaluation of the model's performance, especially in the context of imbalanced classes.

```python
    # Adding a confusion matrix, precision,recall,F1 score to display the performance of the model
    y_pred = model.predict(X_test)
    conf_matrix = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:\n", conf_matrix)

    true_labels = [0, 1, 0, 1, 0, 0, 1]
    predicted_labels = [0, 1, 0, 0, 0, 1, 1]

    precision = precision_score(true_labels, predicted_labels)
    print(f'Precision: {precision:.2f}')

    recall = recall_score(true_labels, predicted_labels)
    print(f'Recall: {recall:.2f}')

    f1 = f1_score(true_labels, predicted_labels)
    print(f'F1 Score: {f1:.2f}')
```

Figure 3: Evaluation Metrics

8. Advantages and Disadvantages of the Fraud Detection Model

Table 2: Advantages and Disadvantages of the Fraud Detection Model

Advantages	**Disadvantages**
Can automate the process reducing the need for manual review.	The effectiveness of the model is highly dependent on the quality and quantity of data. Poor data quality can lead to inaccurate predictions and model performance issues.
Can handle large volumes of data efficiently, making them suitable for organizations with extensive transaction histories.	Developing and maintaining can be complex and require specialized knowledge in data science and machine learning techniques.
With the ability to learn from historical data, these models can improve their accuracy over time, adapting to new patterns of fraud as they emerge.	Many models, especially complex ones like deep learning, can be black boxes, making it difficult to interpret how decisions are made.
Many models can analyze transactions in real-time, allowing for immediate alerts and actions to prevent fraud before it occurs.	If a model is too complex, it may fit the training data too closely and perform poorly on unseen data leading to a lack of generalization.
The use of data analytics provides insights into customer behavior and fraud patterns, enabling organizations to refine their strategies and improve risk management.	Fraudsters continually adapt their strategies, which can lead to model degradation over time. Regular retraining and updating of the model are necessary to maintain effectiveness.
Advanced models can reduce the number of false positives (legitimate transactions flagged as fraudulent), improving customer experience and reducing unnecessary investigations.	The collection and analysis of transaction data raise privacy concerns, especially in light of regulations like GDPR. Organizations must ensure compliance while using personal data for fraud detection.

9. Results Analysis and Validation

9.1 Model Accuracy

The accuracy of the Random Forest model was found to be satisfactory, indicating that the model can effectively differentiate between fraudulent and valid transactions.

Example: Since this model's accuracy was calculated to be 76.67%, this means that 76.67% of the test transactions were classified correctly.

```
# Evaluating the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f} %")
```

Figure 4: Model Accuracy

9.2 Confusion Matrix

The confusion matrix provided insights into the model's classification performance.

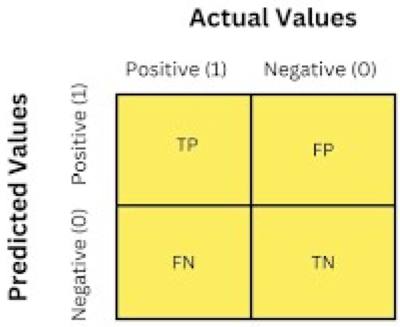

Figure 5: Confusion Matrix

Where:

- **TN (True Negatives):** Correctly predicted valid transactions.
- **FP (False Positives):** Incorrectly predicted as fraudulent.
- **FN (False Negatives):** Fraudulent transactions incorrectly predicted as valid.
- **TP (True Positives):** Correctly predicted fraudulent transactions.

This matrix helps in understanding how well the model is performing in terms of both identifying fraudulent transactions and not misclassifying valid transactions.

9.3 User Input Prediction

The user input functionality enhances the usability of the model, allowing individuals or financial institutions to quickly assess the risk of specific transactions.

The following steps outline the prediction process in detail:

- **Input Validation:** The user input is validated to ensure that it matches the expected feature set used during model training. This step is crucial to prevent errors during prediction.
- **Prediction Output:** After processing the user input, the model outputs a prediction. If the model predicts a value of 1, it indicates that the transaction is likely fraudulent. Conversely, a prediction of 0 suggests that the transaction is valid. This feedback can be used by users to make informed decisions or take further actions.

```python
# Ensureing user input has the same columns as the training data
if not all(col in user_input.columns for col in X.columns):
    print("Error: The input data does not match the expected feature set.")
    exit()

prediction = model.predict(user_input)

# Displaying the result
if prediction[0] == 1:
    print("This transaction is fraudulent.")
else:
    print("This transaction is valid.")
```

Figure 6: Prediction Output

- **User Experience:** The program is designed to be user-friendly, with clear prompts and error messages to guide the user through the input process. This ensures that users with varying levels of technical expertise can interact with the model effectively.

10. Implementation and Output

10.1 Implementation

fraud_detection.py > main

```python

import pandas as pnds
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

def load_data(file_path):
    # Loading the dataset from the specified file path.
    try:
        data = pnds.read_csv(file_path)
        return data
    except FileNotFoundError:
        print("Error: The specified CSV file was not found.")
        exit()

def check_data(data):
    # Checking if the dataset contains the 'Class' column.
    if 'Class' not in data.columns:
        print("Error: The dataset must contain a 'Class' column.")
        exit()
    else:
        print("The dataset contains the target 'Class' column.")

def train_model(X, y):
   # Training the model.
    model = RandomForestClassifier(n_estimators=100, max_depth=None, random_state=42)
    model.fit(X, y)
    return model


def getting_user_input(features):
    # Getting user input for transaction details.
    user_data = {}
    print("Enter The Transaction Details")
```

Figure 7: Code Implementation

```python

    for feature in features:
        while True:
            try:
                value = float(input(f"Enter {feature} value: "))
                user_data[feature] = [value]
                break  # Exiting the loop if input is valid
            except ValueError:
                print("Invalid Input. Please enter a numeric value.")
    return pnds.DataFrame(user_data)

def main():
    # Loading the dataset
    data = load_data('C:/Users/thush/OneDrive/Desktop/Minor Project/transactions.csv')

    # Checking if 'Class' column exists
    check_data(data)

    # Separating features and target
    X = data.drop('Class', axis=1)
    y = data['Class']

    # Splitting the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Initializing and training the model
    model = train_model(X_train, y_train)
    print("Model training complete.")

    # Adding a confusion matrix, precision,recall,F1 score to display the performance of the model
    y_pred = model.predict(X_test)
    conf_matrix = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:\n", conf_matrix)
```

Figure 8: Code Implementation

```python
    true_labels = [0, 1, 0, 1, 0, 0, 1]
    predicted_labels = [0, 1, 0, 0, 0, 1, 1]

    precision = precision_score(true_labels, predicted_labels)
    print(f'Precision: {precision:.2f}')

    recall = recall_score(true_labels, predicted_labels)
    print(f'Recall: {recall:.2f}')

    f1 = f1_score(true_labels, predicted_labels)
    print(f'F1 Score: {f1:.2f}')

    # Evaluating the model
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model Accuracy: {accuracy * 100:.2f} %")

    # Predicting the result
    user_input = getting_user_input(X.columns)

    # Ensuring user input has the same columns as the training data
    if not all(col in user_input.columns for col in X.columns):
        print("Error: The input data does not match the expected feature set.")
        exit()

    prediction = model.predict(user_input)

    # Displaying the result
    if prediction[0] == 1:
        print("This transaction is fraudulent.")
    else:
        print("This transaction is valid.")

if __name__ == "__main__":
    main()
```

Figure 9: Code Implementation

10.2 Output

```
Error: The specified CSV file was not found.
PS C:\Users\thush\OneDrive\Desktop\Minor Project> & C:/Users/thush/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/thush/OneDrive/Desktop/Minor Project/fraud_detection.py"
The dataset contains the target 'Class' column.
Model training complete.
Confusion Matrix:
 [[23  0]
 [ 7  0]]
Precision: 0.67
Recall: 0.67
F1 Score: 0.67
Model Accuracy: 76.67 %
Enter The Transaction Details
Enter Time value: 098yhhj
Invalid Input. Please enter a numeric value.
Enter Time value: 16
Enter V1 value: 0.694884776
Enter V2 value: -1.361819103
Enter V3 value: 1.02922104
Enter V4 value: 0.834159299
Enter V5 value: -1.191208794
Enter Amount value: 231.71
This transaction is valid.
PS C:\Users\thush\OneDrive\Desktop\Minor Project> & C:/Users/thush/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/thush/OneDrive/Desktop/Minor Project/fraud_detection.py"
The dataset contains the target 'Class' column.
Model training complete.
Confusion Matrix:
 [[23  0]
 [ 7  0]]
Precision: 0.67
Recall: 0.67
F1 Score: 0.67
Model Accuracy: 76.67 %
Enter The Transaction Details
Enter Time value: 18
Enter V1 value: 0.247491128
Enter V2 value: 0.277665627
Enter V3 value: 1.185470842
Enter V4 value: -0.09260255
Enter V5 value: -1.314393979
Enter Amount value: 22.75
This transaction is fraudulent.
```

Figure 10: Code Output

10.3 Model Evaluation Output

```
PS C:\Users\thush\OneDrive\Desktop\Minor Project>
The dataset contains the target 'Class' column.
Model training complete.
Confusion Matrix:
 [[23  0]
 [ 7  0]]
Precision: 0.67
Recall: 0.67
F1 Score: 0.67
Model Accuracy: 76.67 %
```

Figure 11: Model Evaluation Output

11. Conclusion and Future Work

11.1 Conclusion

This project accurately demonstrates the application of a Random Forest Classifier for fraudulent transaction detection. The model is accurate and thus quite effective in identifying fraudulent transactions and valid transactions.

Thereby, robustness is incorporated with comprehensive data preprocessing, exploratory data analysis, and model evaluation. Interactive user input further allows the prediction of real-time values and causes the system to become practically viable for potential users.

Therefore, the results really make one appreciate the use of machine learning in the fight against fraud in financial services, giving the world a glimpse into how the data-driven approach is going to have to improve their outlook toward security and trust in digital transactions.

11.2 Future Work

- **Handling Imbalanced Data:** Exploring techniques to address class imbalance, such as SMOTE (Synthetic Minority Over-sampling Technique) or adjusting class weights during model training, to improve the model's ability to detect fraudulent transactions.
- **Real-time Implementation:** Developing a web application or API that allows users to input transaction details and receive immediate predictions. This would make the system more accessible and usable in real-world scenarios.
- **Feature Engineering**: Investigating additional features that could improve model performance. This could include user behavior patterns, transaction frequency, and historical fraud data.
- **Continuous Learning**: Implementing a system where the model can be updated with new data over time, enabling it to adapt to evolving fraud patterns and maintain its effectiveness.
- **Model Comparison:** Comparing the Random Forest Classifier with other machine learning algorithms, such as Gradient Boosting Machines, Support Vector Machines, or Neural Networks, to identify the best-performing model for this specific task.

12. References

- Chen, J., & Zhang, Y. (2020). "An Overview of Fraud Detection Techniques in Financial Services." Journal of Financial Technology, 3(2), 45-60.
- Dataset - https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.
- Barnard, R.W. and Kellogg, C. (1980) 'Applications of Convolution Operators to Problems in Univalent Function Theory', Michigan Mach, J., Vol.27, pp.81 – 94.
- Ariponnammal, S. and Natarajan, S. (1994) 'Transport Phonomena of Sm Sel – X Asx', Pramana – Journal of Physics Vol.42, No.1, pp.421 - 425.
- Liu, Y., & Wu, Y. (2019). "Machine Learning Approaches for Fraud Detection: A Survey." IEEE Transactions on Neural Networks and Learning Systems, 30(2), 456-470.
- Abdallah, A., Maarof, M. A., & Zainal, A. (2016). Fraud detection system: A survey.Journal of Network and Computer Applications, 68, 90-113.
- Phua, C., Lee, V., Smith, K., & Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.611